Advances in Generative Adversarial Networks

GANs

Maciej Żelaszczyk

June 5, 2019

PhD Student in Computer Science
Division of Artificial Intelligence and Computational Methods
Faculty of Mathematics and Information Science

m.zelaszczyk@mini.pw.edu.pl

Warsaw University of Technology

Supervised vs. unsupervised

Supervised:

- Requires huge datasets.
- Annotating is costly.
- Extensive training.
- Driving a car off a cliff.
- Learns tasks, not skills.
- Some well-specified tasks have been largely solved.
- Limit to how much data we can obtain.
- Ignores physical world.

Supervised vs. unsupervised

How do children learn?

- A lot of evolutionary knowledge.
- Vision, hearing, touch etc. in place.
- Extensive observation.
- Build a model of the world.
- Model vs. physical world.
- Surprise, curiosity guide learning.
- Continuous refinement of model.
- Limited reinforcement learning.
- All initial learning is unsupervised.

Supervised vs. unsupervised

Unsupervised:

- In practice, very little labelled data available.
- Need to create model of world, confront it with reality.
- Attend to data.
- Manipulate world.
- Learn from little external reward.
- Learn from very few examples.
- Exploit physical structure of world to obtain links.
- Learn skills rather than tasks.

Importance of unsupervised learning

What if importance of various kinds of learning is like a cake?

- Pure reinforcement learning = cherry.
- Supervised learning = icing.
- Unsupervised/self-supervised/predictive learning = génoise.
- Perhaps we are still missing a sizeable pie crust? = meta-learning.

Source: LeCun, Y., *The Next Step Towards Artificial Intelligence*

Generative models

Models:

- Discriminative: $P(Y|X = x)$
- Generative. Joint probability distribution: $X \times Y, P(X, Y)$
- No hard demarcation line.

Standard generative models in deep learning:

- Autoencoders.
- Variational autoencoders (VAEs).
- Generative adversarial networks (GANs).

Generative Adversarial Nets

Approach model training from game-theoretic point of view [Goodfellow et al., 2014]:

- Two networks: Generator and Discriminator.
- Generator: from latent variable $\mathbf{z}$ generate into data space.
- Discriminator: distinguish between real and generated data.
- Generator tries to "fool" the Discriminator.
- Discriminator strives to "look through" the Discriminator.
- This can be represented by a minimax two-player game.

Generative Adversarial Nets

More concretely:

- We aim to learn Generator's distribution p_g over data $\mathbf{x}$.
- Define prior $p_{\mathbf{z}}(\mathbf{z})$.
- Represent mapping to data space $G(\mathbf{z}; \theta_g)$.
- G is a neural network parametrized by θ_g.
- Define second neural network $D(\mathbf{x}; \theta_d)$ which outputs single scalar.
- $D(\mathbf{x})$ represents a probability that $\mathbf{x}$ came from the data rather than p_g.

Source: [Goodfellow et al., 2014]

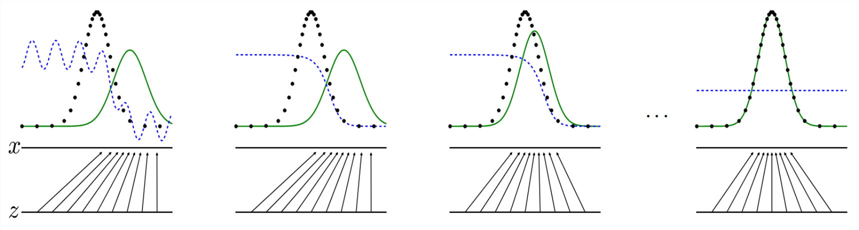
Generative Adversarial Nets

Training:

- Train D to maximize probability of assigning correct label to real data and samples from G.
- Train G to maximize probability of D assigning incorrect label to samples from G.
- D and G play:
- $\min_G \max_D V(D, G) = \mathbb{E}_{\mathbf{x} \sim p_{data}(\mathbf{x})}[\log(D(\mathbf{x}))] + \mathbb{E}_{\mathbf{z} \sim p_{\mathbf{z}}(\mathbf{z})}[\log(1 - D(G(\mathbf{z})))]$.
- $\log(1 - D(G(\mathbf{z})))$ may saturate early in training.
- Can train G to maximize $\log(D(G(\mathbf{z})))$ instead.

Source: [Goodfellow et al., 2014]

Generative Adversarial Nets



Source: [Goodfellow et al., 2014]

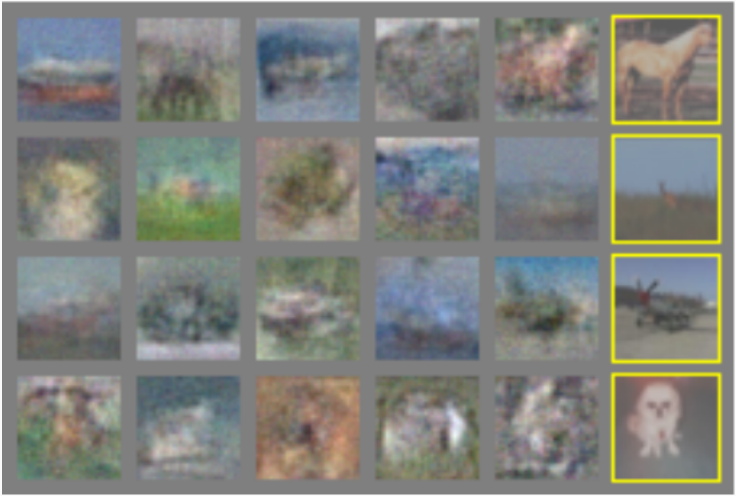
Generative Adversarial Nets

Source: [Goodfellow et al., 2014]

Generative Adversarial Nets

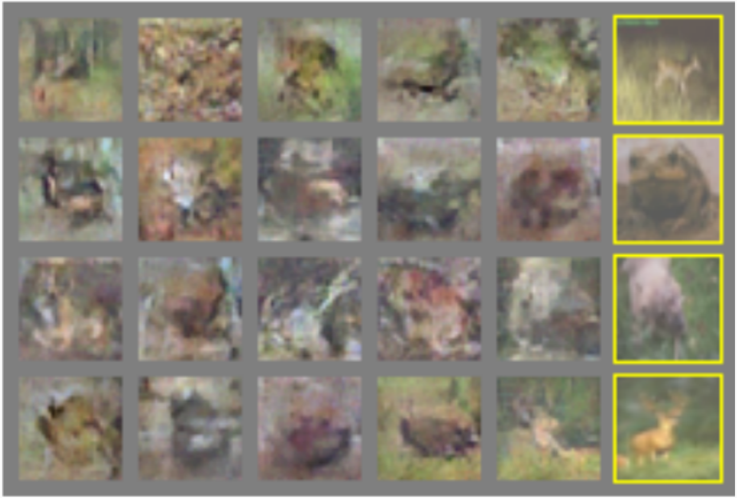
Source: [Goodfellow et al., 2014]

Generative Adversarial Nets



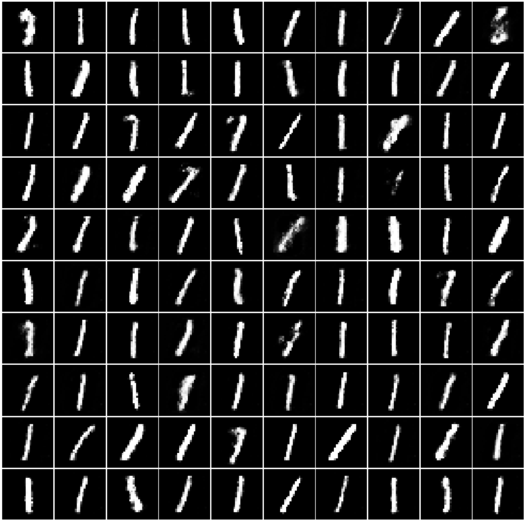
Source: [Goodfellow et al., 2014]

Generative Adversarial Nets



Source: [Goodfellow et al., 2014]

Mode collapse



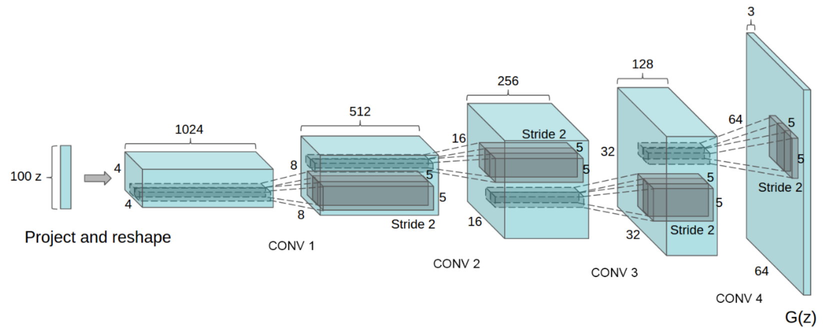
Deep Convolutional GAN

Architecture guidelines for Deep Convolutional GANs (DCGAN):

- Replace any pooling layers with strided convolutions (Discriminator) and fractional-strided convolutions (Generator).
- Use batchnorm in both the generator and the discriminator.
- Remove fully connected hidden layers for deeper architectures.
- Use ReLU activation in generator for all layers except for the output, which uses Tanh.
- Use LeakyReLU activation in the discriminator for all layers.

Source: [Radford et al., 2016]

Deep Convolutional GAN



Source: [Radford et al., 2016]

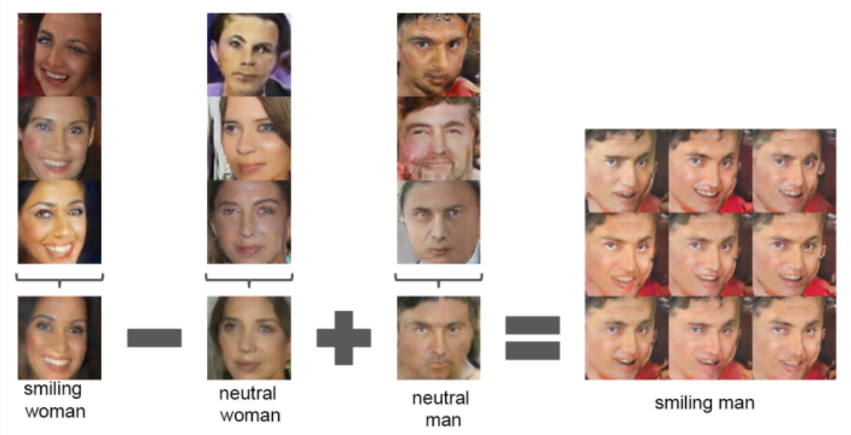
Deep Convolutional GAN

Source: [Radford et al., 2016]

Deep Convolutional GAN

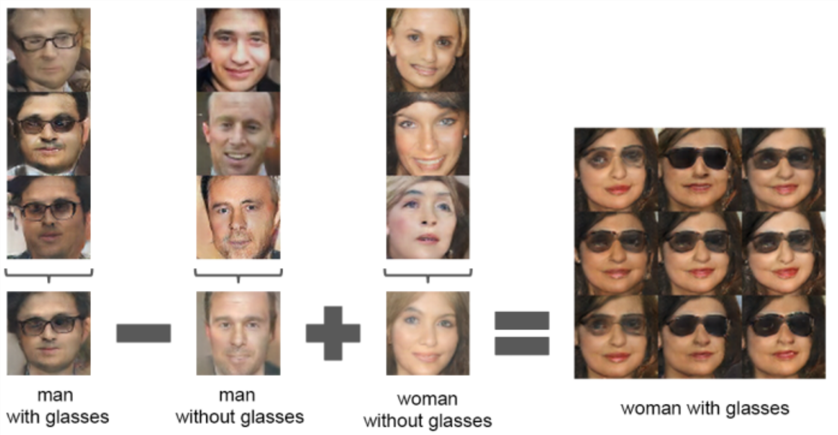
Source: [Radford et al., 2016]

Deep Convolutional GAN



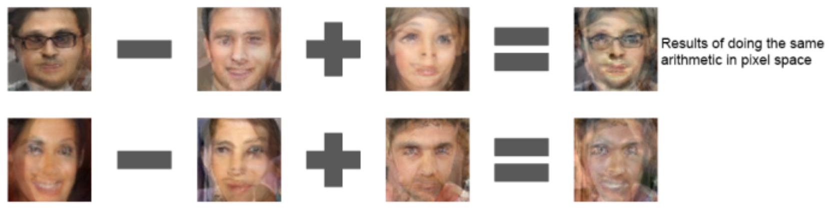
Source: [Radford et al., 2016]

Deep Convolutional GAN



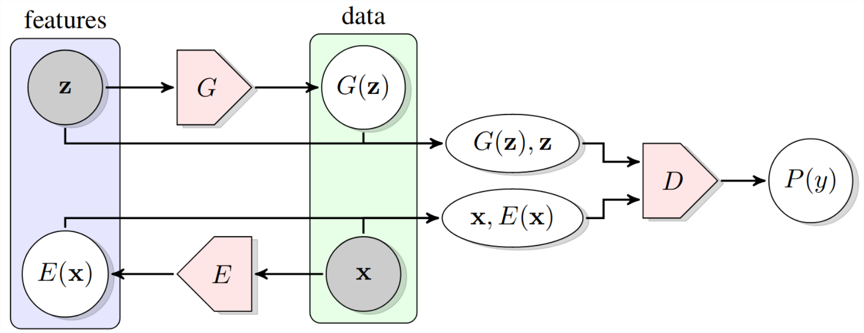
Source: [Radford et al., 2016]

Deep Convolutional GAN



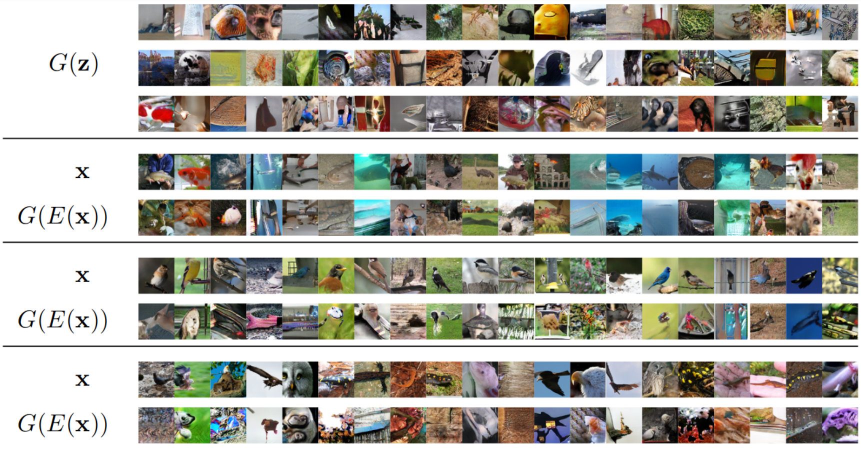
Source: [Radford et al., 2016]

Bidirectional GAN



Source: [Donahue et al., 2017]

Bidirectional GAN



Source: [Donahue et al., 2017]

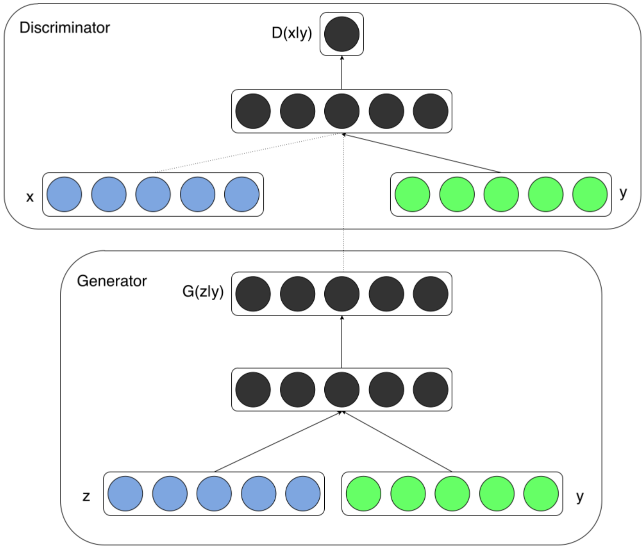
Conditional Generative Adversarial Nets

Classic GAN training can be reformulated to incorporate additional knowledge:

- D and G play:
- $\min\limits_{G} \max\limits_{D} V(D, G) = \mathbb{E}_{\mathbf{x} \sim p_{data}(\mathbf{x})}[\log(D(\mathbf{x}|\mathbf{c}))] + \mathbb{E}_{\mathbf{z} \sim p_{\mathbf{z}}(\mathbf{z})}[\log(1 - D(G(\mathbf{z}|\mathbf{c})))]$.
- Training G to maximize $\log(D(G(\mathbf{z}|\mathbf{c})))$ still works.

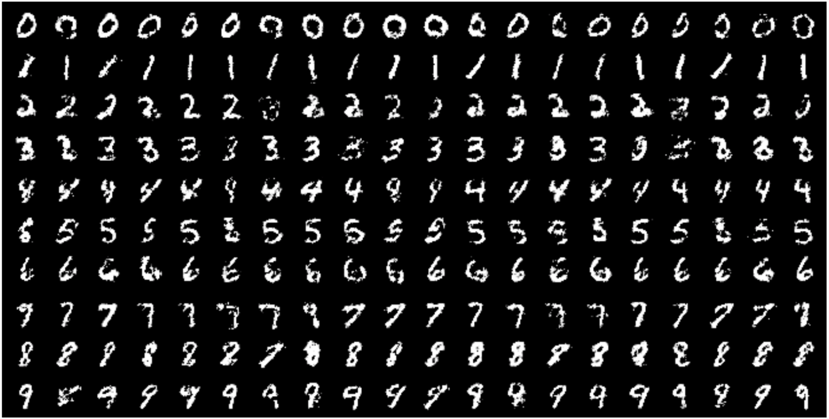
Source: [Mirza and Osindero, 2014]

Conditional Generative Adversarial Nets



Source: [Mirza and Osindero, 2014]

Conditional Generative Adversarial Nets



Source: [Mirza and Osindero, 2014]

Information Maximizing Generative Adversarial Nets

In traditional GANs, G is not restricted. Representations can be disentangled but there is no such guarantee. Possible to use more structure without sacrificing unsupervised training:

- Many domains naturally decompose into a set of semantically meaningful factors of variation.
- MNIST example: allocate a discrete random variable to represent the digit (0-9), choose to have two additional continuous variables representing the digits angle and thickness of the digits stroke.
- Decompose the input noise vector into two parts: (i) $\mathbf{z}$, which is treated as source of incompressible noise; (ii) latent code $\mathbf{c}$, the salient structured semantic features of the data distribution.

Source: [Chen et al., 2016]

Information Maximizing Generative Adversarial Nets

Possible to introduce additional constraints:

- Generator becomes $G(\mathbf{z}, \mathbf{c})$.
- In standard GAN, the generator is free to ignore the additional latent codes by finding a solution satisfying $P_G(\mathbf{z}|\mathbf{c}) = P_G(\mathbf{x})$
- To cope with trivial codes, introduce information-theoretic regularization: there should be high mutual information between latent codes $\mathbf{c}$ and generator distribution $G(\mathbf{z}, \mathbf{c})$.
- $I(\mathbf{c}; G(\mathbf{z}, \mathbf{c}))$ should be high.
- In information theory, mutual information between X and Y, $I(X; Y)$, measures the amount of information learned from knowledge of random variable Y about the other random variable X.
- $I(X; Y) = H(X) - H(X|Y) = H(Y) - H(Y|X)$.

Source: [Chen et al., 2016]

Information Maximizing Generative Adversarial Nets

Information-regularized minimax game:

- $\min_G \max_D V_I(D, G) = V(D, G) - \lambda I(\mathbf{c}; G(\mathbf{z}, \mathbf{c}))$.
- In practice, $I(\mathbf{c}; G(\mathbf{z}, \mathbf{c}))$ is hard to maximize directly.
- Variational Information Maximization.
- Define a variational lower bound.
- $L_I(G, Q) = \mathbb{E}_{\mathbf{c}\sim P(\mathbf{c}), \mathbf{x}\sim G(\mathbf{z},\mathbf{c})} [\log(Q(\mathbf{c}|\mathbf{x}))] + H(\mathbf{c}) =$ $\mathbb{E}_{\mathbf{x}\sim G(\mathbf{z},\mathbf{c})} \left[\mathbb{E}_{\mathbf{c}'\sim P(\mathbf{c}|\mathbf{x})} \left[\log(Q(\mathbf{c}'|\mathbf{x}))\right]\right] + H(\mathbf{c}) \leq I(\mathbf{c}; G(\mathbf{z}, \mathbf{c}))$
- $\min_{G,Q} \max_D V_{\text{InfoGAN}}(D, G, Q) = V(D, G) - \lambda L_I(G, Q)$.

Source: [Chen et al., 2016]

Information Maximizing Generative Adversarial Nets

Implementation:

- Parametrize the auxiliary distribution Q as a neural network.
- Q and D share all convolutional layers and there is one final fully connected layer to output parameters for the conditional distribution $Q(\mathbf{c}|\mathbf{x})$.
- InfoGAN only adds a negligible computation cost to GAN.
- $L_I(G, Q)$ "always" converges faster than normal GAN objectives.
- InfoGAN essentially comes for "free" with GAN.

Source: [Chen et al., 2016]

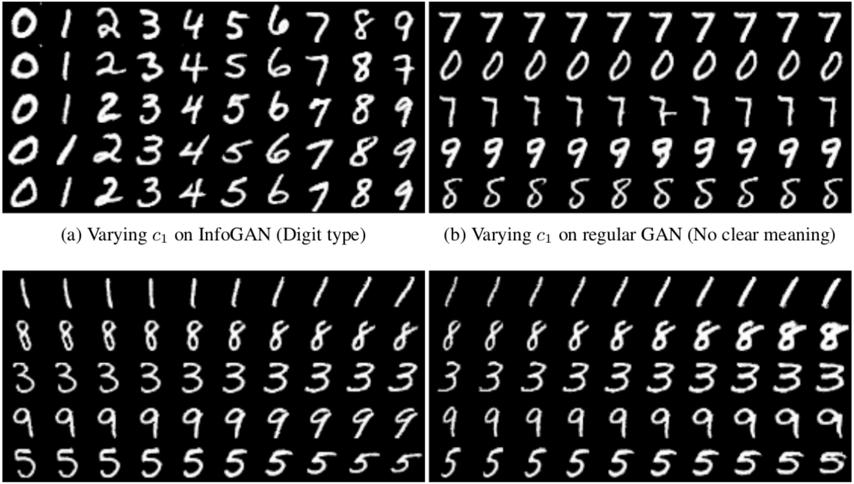
Information Maximizing Generative Adversarial Nets

Implementation:

- For categorical latent code c_i, softmax nonlinearity to represent $Q(c_i|\mathbf{x})$.
- For continuous latent code c_j, more options depending on the true posterior $P(c_j|\mathbf{x})$. Treating $Q(c_j|\mathbf{x})$ as a factored Gaussian seems sufficient.
- For categorical latent code $\lambda = 1$ sufficient.
- For continuous latent code smaller values of λ.

Source: [Chen et al., 2016]

Information Maximizing Generative Adversarial Nets



(a) Varying c_1 on InfoGAN (Digit type)

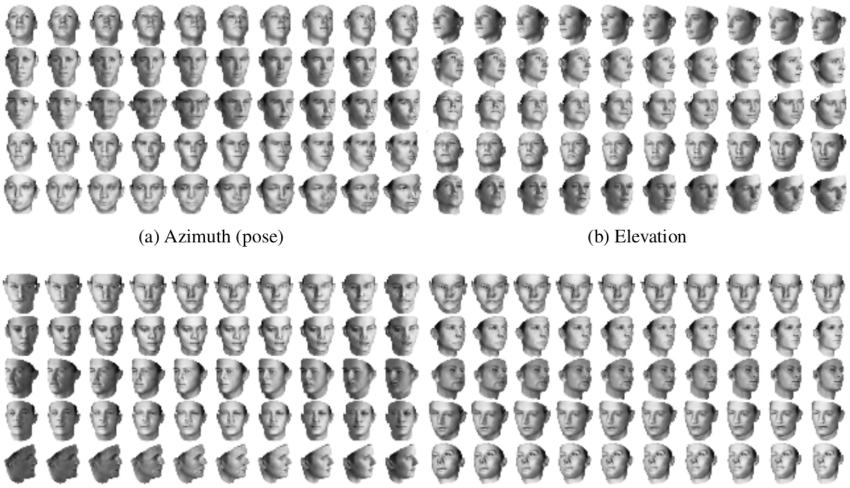
(b) Varying c_1 on regular GAN (No clear meaning)

(c) Varying c_2 from -2 to 2 on InfoGAN (Rotation)

(d) Varying c_3 from -2 to 2 on InfoGAN (Width)

Source: [Chen et al., 2016]

Information Maximizing Generative Adversarial Nets



(a) Azimuth (pose) (b) Elevation

(c) Lighting (d) Wide or Narrow

Source: [Chen et al., 2016]

Information Maximizing Generative Adversarial Nets

(a) Rotation (b) Width

Source: [Chen et al., 2016]

Information Maximizing Generative Adversarial Nets

(a) Azimuth (pose)

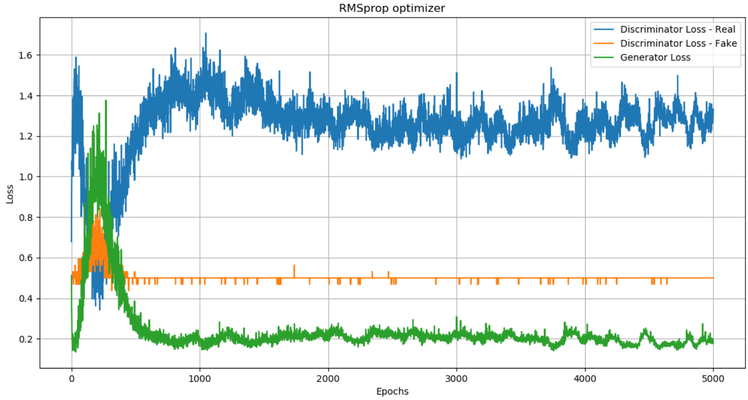
(b) Presence or absence of glasses

(c) Hair style

(d) Emotion

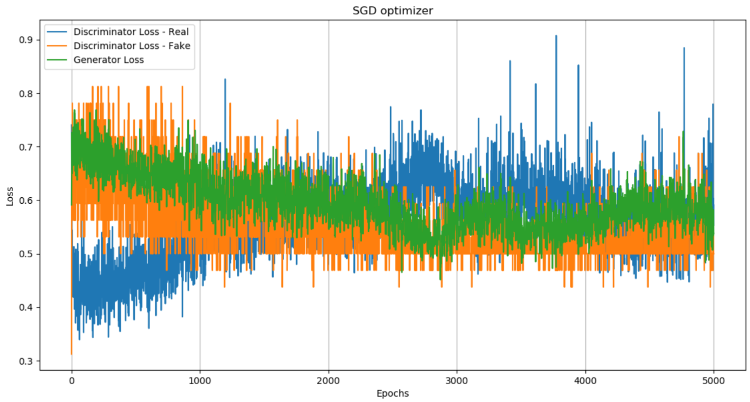
Source: [Chen et al., 2016]

Loss functions



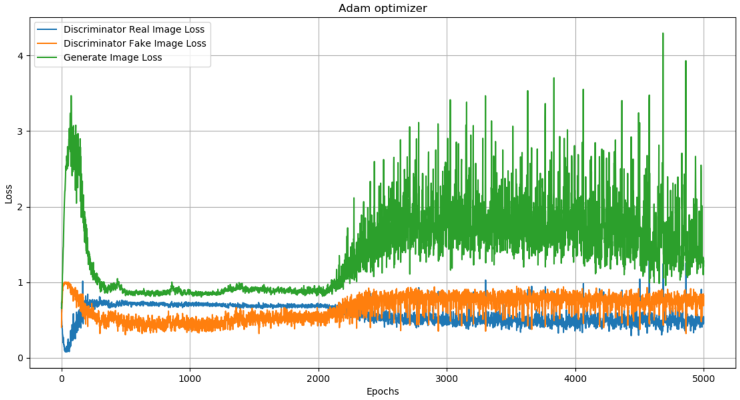
Source: Jayathilaka, M. *Understanding and optimizing GANs (Going back to first principles)*

Loss functions



Source: Jayathilaka, M. *Understanding and optimizing GANs (Going back to first principles)*

Loss functions



Source: Jayathilaka, M. *Understanding and optimizing GANs (Going back to first principles)*

Jensen-Shannon Divergence

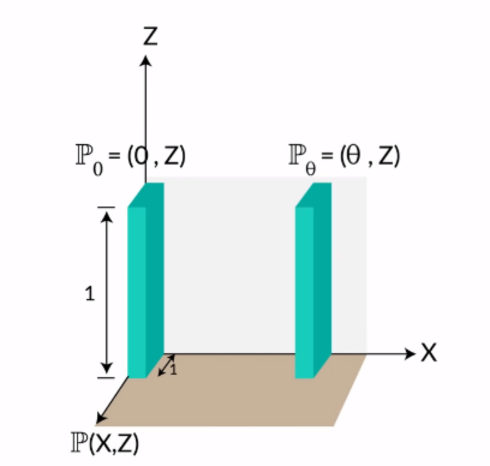
In the original GAN, the objective for the discriminator can be reforulated as follows:

- $C(G) = \max_D V(G, D) = -\log(4) + 2JSD(p_{data} \| p_g)$.
- JSD is the Jensen-Shannon Divergence.
- The global minimum of $C(G)$ is $C^* = -\log(4) + 2JSD(p_{data} \| p_g)$.
- The only solution is $p_g = p_{data}$.
- There is a serious problem with JSD.

Source: [Goodfellow et al., 2014]

Jensen-Shannon Divergence

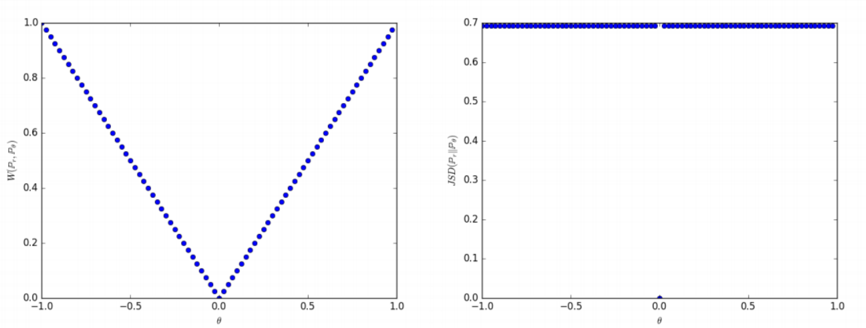
Even for very simple distributions, $\theta \to 0$ does not guarantee $JSD(\mathbb{P}_\theta, \mathbb{P}_0) \to 0$:



Source: Midnight math stories, *Nuts and Bolts of WGANs, Kantorovich-Rubistein Duality, Earth Movers Distance*

Jensen-Shannon Divergence

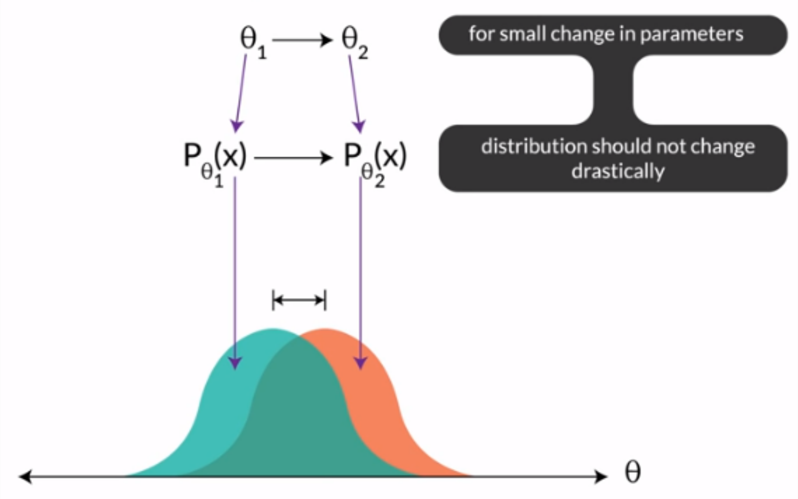
There might not be enough gradient to facilitate learning.



Source: [Arjovsky et al., 2017]

Jensen-Shannon Divergence

In other words:



Source: Midnight math stories, *Nuts and Bolts of WGANs, Kantorovich-Rubistein Duality, Earth Movers Distance*

Wasserstein Generative Adversarial Networks

Basic idea, use a different metric to define distance between probabilities:

- Earth-Mover (EM) distance or Wasserstein-1.
- $W(\mathbb{P}_r, \mathbb{P}_g) = \inf_{\gamma \in \prod(\mathbb{P}_r, \mathbb{P}_g)} \mathbb{E}_{(x,y)\sim\gamma}\left[\|x - y\|\right]$
- $\prod(\mathbb{P}_r, \mathbb{P}_g)$ denotes the set of all joint distributions $\gamma(x, y)$ whose marginals are respectively $\mathbb{P}_r$ and $\mathbb{P}_g$.
- Intuitively, $\gamma(x, y)$ indicates how much mass must be transported from x to y in order to transform the distributions $\mathbb{P}_r$ into the distribution $\mathbb{P}_g$. The EM distance then is the cost of the optimal transport plan.

Source: [Arjovsky et al., 2017]

Wasserstein Generative Adversarial Networks

Algorithm 1 WGAN, our proposed algorithm. All experiments in the paper used the default values $\alpha = 0.00005$, $c = 0.01$, $m = 64$, $n_{\text{critic}} = 5$.

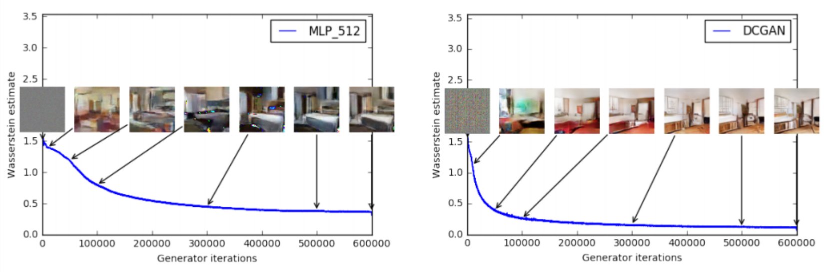
Require: : α, the learning rate. c, the clipping parameter. m, the batch size. n_{critic}, the number of iterations of the critic per generator iteration.

Require: : w_0, initial critic parameters. θ_0, initial generator's parameters.

while θ has not converged **do**
  for $t = 0, ..., n_{\text{critic}}$ **do**
    Sample $\{x^{(i)}\}_{i=1}^{m} \sim \mathbb{P}_r$ a batch from the real data.
    Sample $\{z^{(i)}\}_{i=1}^{m} \sim p(z)$ a batch of prior samples.
    $g_w \leftarrow \nabla_w \left[\frac{1}{m}\sum_{i=1}^{m} f_w(x^{(i)}) - \frac{1}{m}\sum_{i=1}^{m} f_w(g_\theta(z^{(i)}))\right]$
    $w \leftarrow w + \alpha \cdot \text{RMSProp}(w, g_w)$
    $w \leftarrow \text{clip}(w, -c, c)$
  end for
  Sample $\{z^{(i)}\}_{i=1}^{m} \sim p(z)$ a batch of prior samples.
  $g_\theta \leftarrow -\nabla_\theta \frac{1}{m}\sum_{i=1}^{m} f_w(g_\theta(z^{(i)}))$
  $\theta \leftarrow \theta - \alpha \cdot \text{RMSProp}(\theta, g_\theta)$
end while

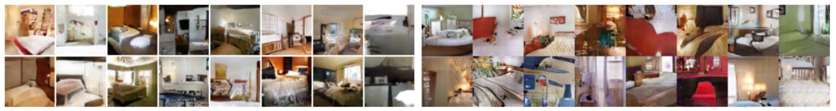
Source: [Arjovsky et al., 2017]

Wasserstein Generative Adversarial Networks



Source: [Arjovsky et al., 2017]

Wasserstein Generative Adversarial Networks



Source: [Arjovsky et al., 2017]

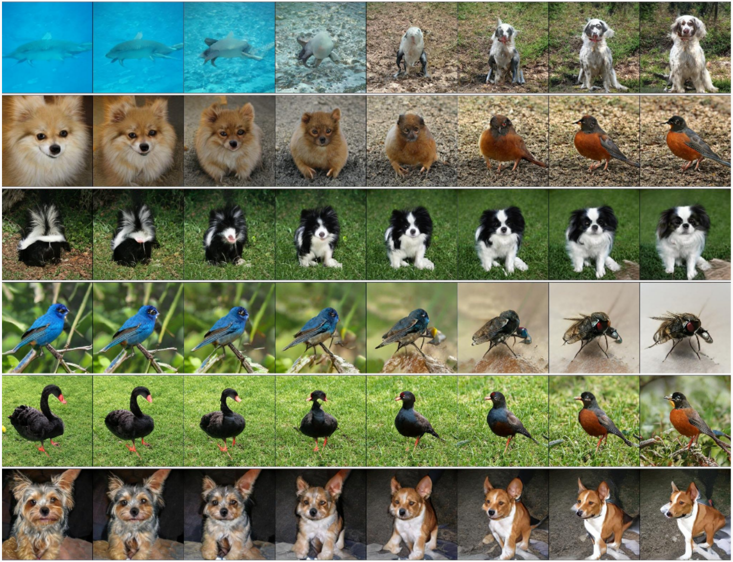
BigGAN

Source: [Brock et al., 2018]

BigGAN

Source: [Brock et al., 2018]

BigGAN



Source: [Brock et al., 2018]

Arjovsky, M., Chintala, S., and Bottou, L. (2017).
Wasserstein gan.
ICML.

Brock, A., Donahue, J., and Simonyan, K. (2018).
Large scale gan training for high fidelity natural image synthesis.
arXiv.

Chen, X., Duan, Y., Houthooft, R., Schulman, J., Sutskever, I., and Abbeel, P. (2016).
Infogan: Interpretable representation learning by information maximizing generative adversarial nets.
NIPS.

Donahue, J., Krhenbhl, P., and Darrell, T. (2017).
Adversarial feature learning.
ICLR.

Goodfellow, I. J., Pouget-Abadie, J., et al. (2014).
Generative adversarial networks.
NIPS.

Mirza, M. and Osindero, S. (2014).
Conditional generative adversarial nets.
arXiv.

Radford, A., Metz, L., and Chintala, S. (2016).
Unsupervised representation learning with deep convolutional generative adversarial networks.
ICLR.